(CS771A) INTRODUCTION TO MACHINE LEARNING

Instructor: Dr. Piyush Rai

Content Based Music Genre Classification using Temporal and Spectral Features

Presented By:

(20111016) Debanjan Chatterjee
(20111407) Gaurav Tank
(20111032) Mayank Bansal

Motivation:

Almost everyone listens to music. It propagates feelings and thoughts between people. Today's music corpus is very heterogenous in nature. Everybody has their own music taste mainly because of the diversity of composers and musicians out there.

Many music streaming services like Spotify, Amazon Music, Pandora etc. uses a state-of-the-art (SOTA) algorithms to find similarities and patterns in the music tracks and based on that, tracks are categorized into different classes known as 'Genres' of the tracks. Based on the genres one listen to, the algorithm can recommend related songs.

As music listeners, we are motivated to get insight into some of the existing music classification techniques and try out experiments using different machine learning algorithms and see if we can improve any of these existing techniques.

Industrial Use:

Several companies uses music classification to segment the database (of songs) according to genre and either, recommend songs accordingly to their users (like done by Spotify, YouTube Music etc.) , or even as a product (like Shazam).

As there are numerous machine learning techniques that work well on extracting patterns, trends and other useful information from a large set of data, these techniques are used in the industry to perform music analysis.

Source – google

Other Approaches:

Besides the content-based music genre classification, other techniques exists as well such as :-

1. Collaborative filtering – This approach makes a prediction about the taste of a user with the help of part of the community that shares the same (or similar) taste (thus called collaborative). An important assumption is made that if a user say A listens to the same songs of a genre as the users B, C, D and E, then A would also prefer the songs of other genres listened to by B, C, D and E.

Drawback – For the collaborative filtering approach to work, there must be a large set of users (i.e. the community) and user data.

2. Knowledge-based – This approach draws in user interests and feedback at regular periods. This technique is used mainly when the other two (content-based and collaborative filtering) cannot be applied.

Drawback – This approach is too much dependent on user feedback (an example is the like and dislike option provided by Spotify for each song). Thus, inadequate feedbacks or unable to obtain feedbacks regularly hinders the working of the approach.

Our Approach: Content-based

This approach focuses on *content* or item similarity. It analyses the past interactions of the user (in our context, the songs that the user listened to) and if the interactions were positive, then recommends similar items to the user.

Though this approach leaves a little room to surprise the user with an out of the box suggestion, but it still works well.

As this approach does not rely on community and also, does not involve the use of feedbacks, thus it overcomes the drawback of both collaborative-filtering and knowledge-based filtering approaches.

About the Dataset

The dataset used in the project is the GTZAN dataset available at the MARYSAS website. This dataset was originally used in [1]. The dataset is made up of about 1000 audio tracks each of which is 30 seconds long. There are about 10 genres, each covering up 100 tracks for each genre. All tracks has 22050 Hz sampling rate, mono channel with 16-bit sample audio files in .wav format.

The audio files were gathered using different sources like CDs, radio, microphone recordings etc. to represent a variety of recording conditions.

Problem Formulation: Music Genre Classification as Time Series Classification.

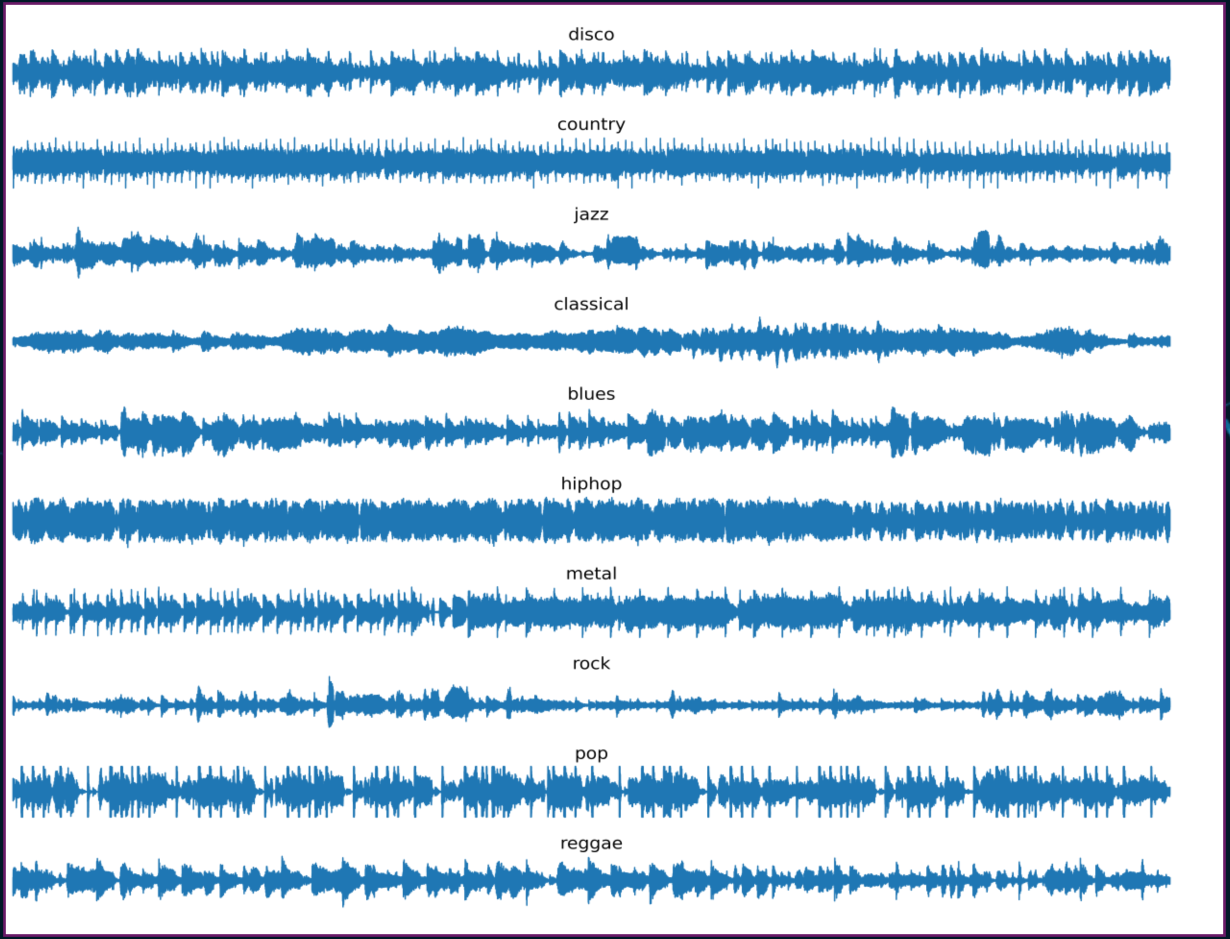


Figure: Waveform plot of randomly selected one song for each genre from dataset.

The figure represents sample waveforms from the different music genres. As we can see that the waveforms are complex, and for it to be interpreted in a meaningful way, by a system, several audio signal processing steps, need to be applied, in order to prepare the dataset on which a classifier model can be learnt. However, from the figure, it can be intuitively observed the genres can be differentiated based on the waveforms

Audio Representation in Computer Memory:

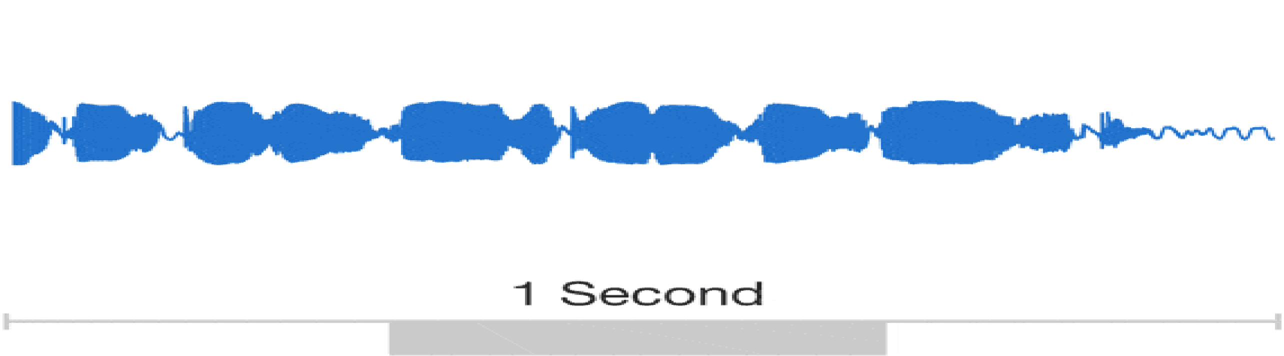


Figure: Taken from WaveNet blog by deepmind.

In computer memory, audio track is represented as arrays of samples where each array represents an audio channel and each sample in an audio channel can be an instance of double, float or signed integer variable depending on the audio depth quality we want to retain.

Mono audio tracks have one array, stereo audio tracks have two arrays. There can be more than two channels too.

GTZAN has only mono channel tracks.

How many samples for 1 second audio? (Sample Rate)

Well, It depends on the sampling rate.

For GTZAN dataset, each audio tracks has sample rate (sr) = 22050 Hz. (meaning: 22050 array elements per second)

Fine Grained Audio Features:

- Amplitude
- Frequency
- Phase Angle
- Pitch(Related to Frequency)
- Tonal Features
- Amplitude Envelope
- Power of Signal

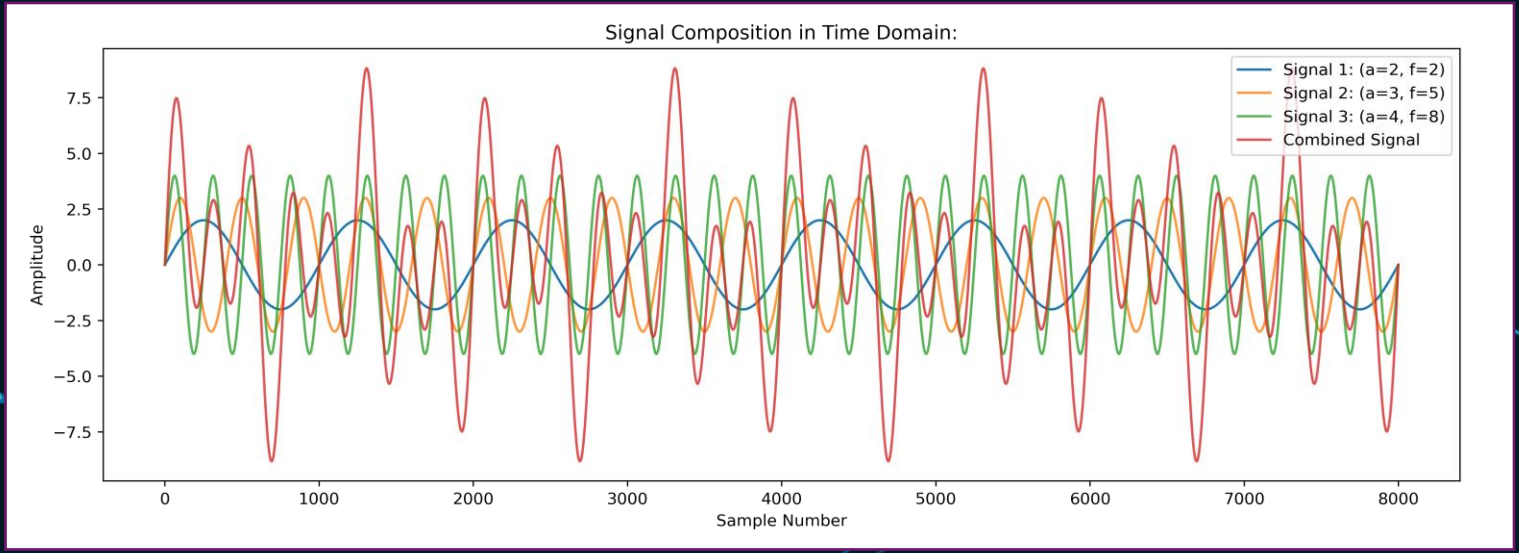
Audio Representation In Different Domains:

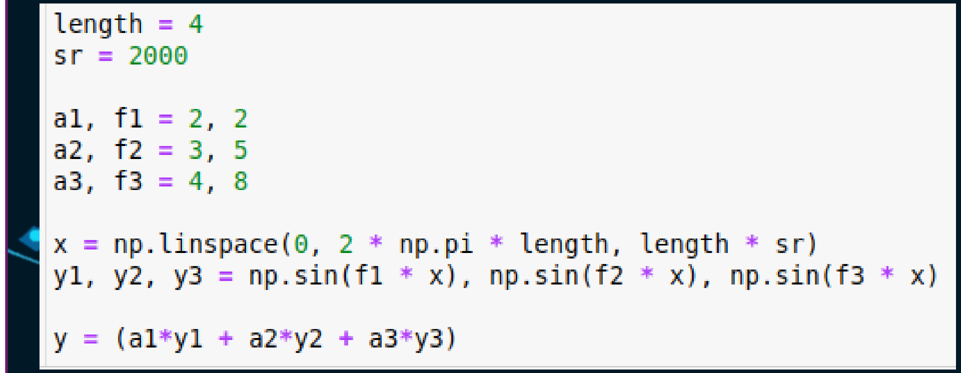
Time Domain (Time vs Amplitude Plot)

Frequency Domain (Frequency vs Amplitude Plot)

Signal (Audio) Representation In Time Domain:

NOTE: In this figure, sample rate is 2000 Hz.



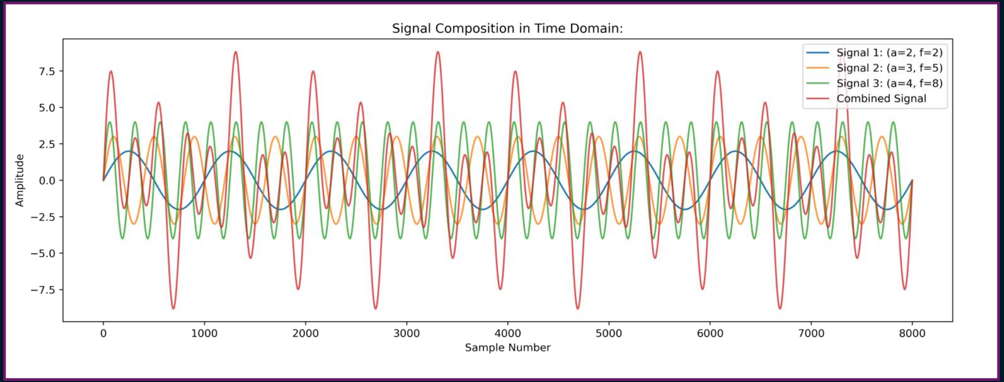


```
length = 4
sr = 2000

a1, f1 = 2, 2
a2, f2 = 3, 5
a3, f3 = 4, 8

x = np.linspace(0, 2 * np.pi * length, length * sr)
y1, y2, y3 = np.sin(f1 * x), np.sin(f2 * x), np.sin(f3 * x)

y = (a1*y1 + a2*y2 + a3*y3)
```

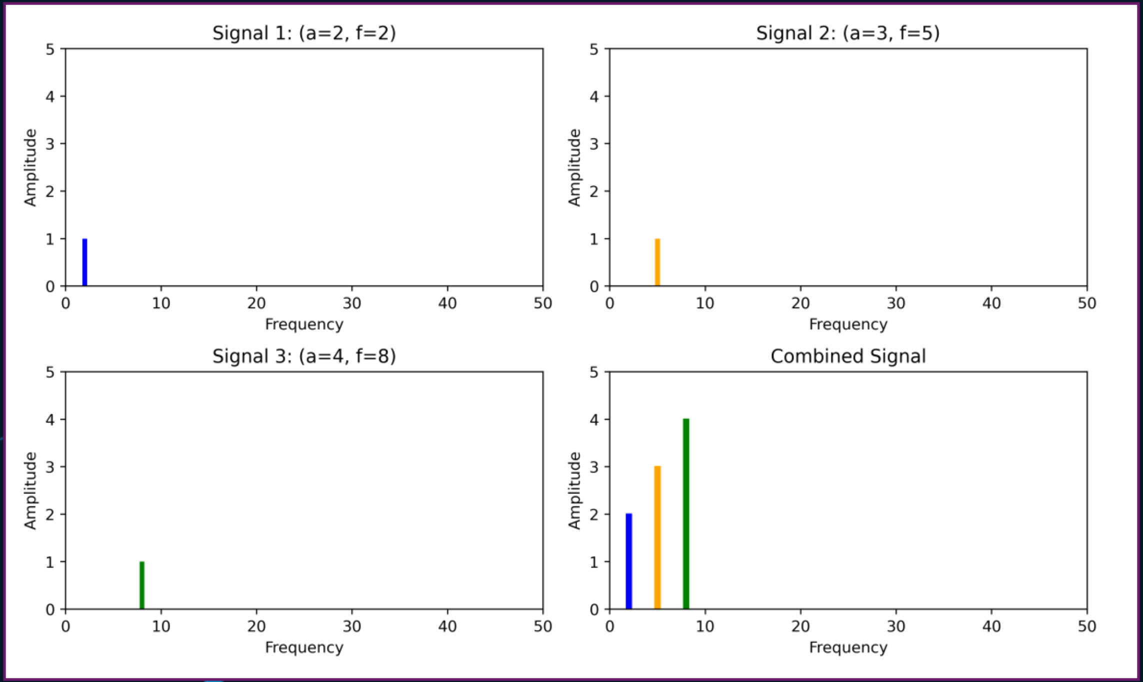
When two or more signals are added, the resultant signal is very tangled, and it is hard to decompose it in time domain.

But hopefully, we can approximate the complex signal by decomposing it in the frequency domain.

Audio Representation In Frequency Domain:



FFT



FFT NumPy code:

```
y_fft = (2 / sr) * np.fft.fft(y, n=sr)
```

But this captures the global features !!!

Using Fourier transform of a signal, we can get its frequency domain representation.

Decomposition is very easy in frequency domain compared to time domain.

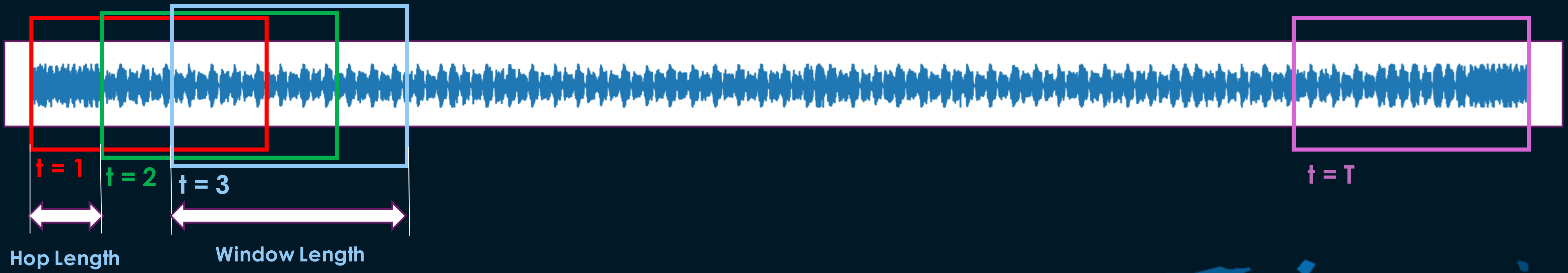
NumPy provides an FFT implementation which is optimized.

In Fourier Transform, we get similarity of our signal with complex signal with different frequencies. And quantification of this similarity gives the amplitude in the frequency domain.

Similarity is given by:

{ 2 * dot(exp(-i x 2πft), Y) } / N

Windowed Feature Extraction of Signals:



t = 1

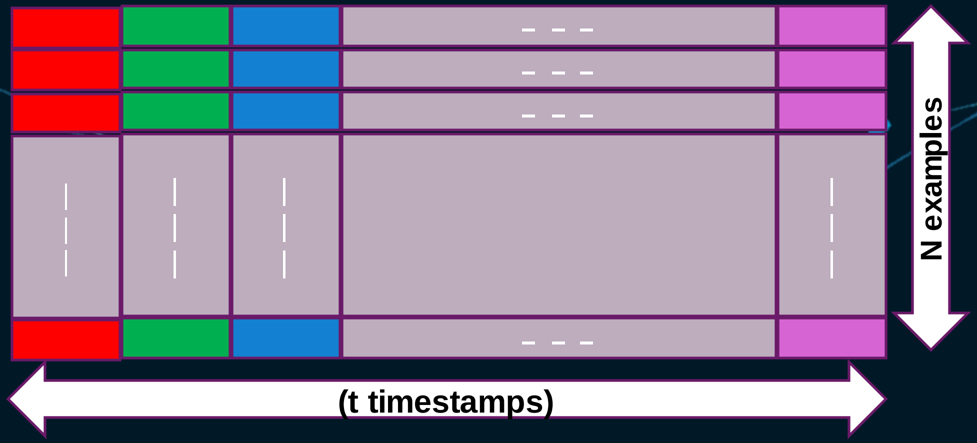
t = 2

t = 3

t = T

Hop Length

Window Length



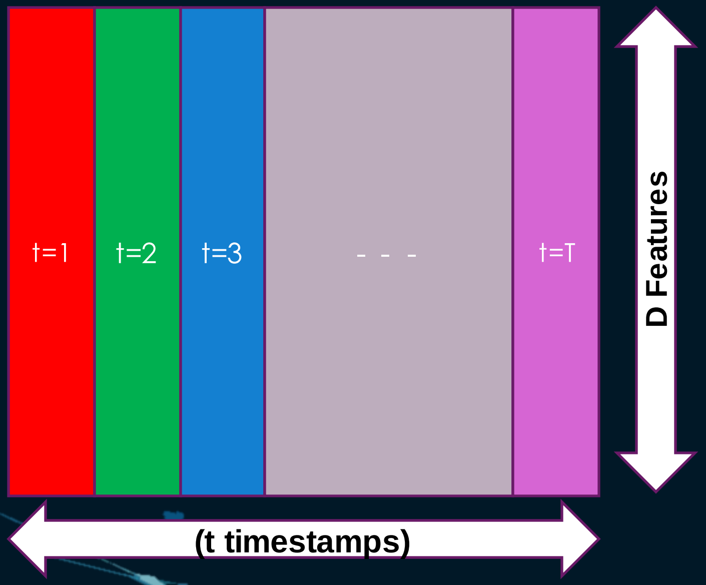
N examples

(t timestamps)

We are mostly interested into finding non-stationary patterns in the audio signals. As simple FFT gives information about whole track, it finds the global features. But, with some modification of FFT will give us very nice temporal features!

Modification: Find the local features by windowing method with stride equal to Hop Length.

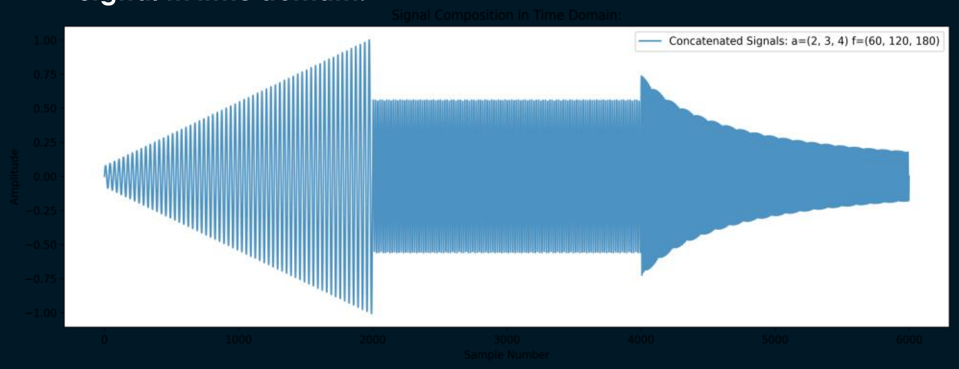
Spectrogram: Array of local FFT features.



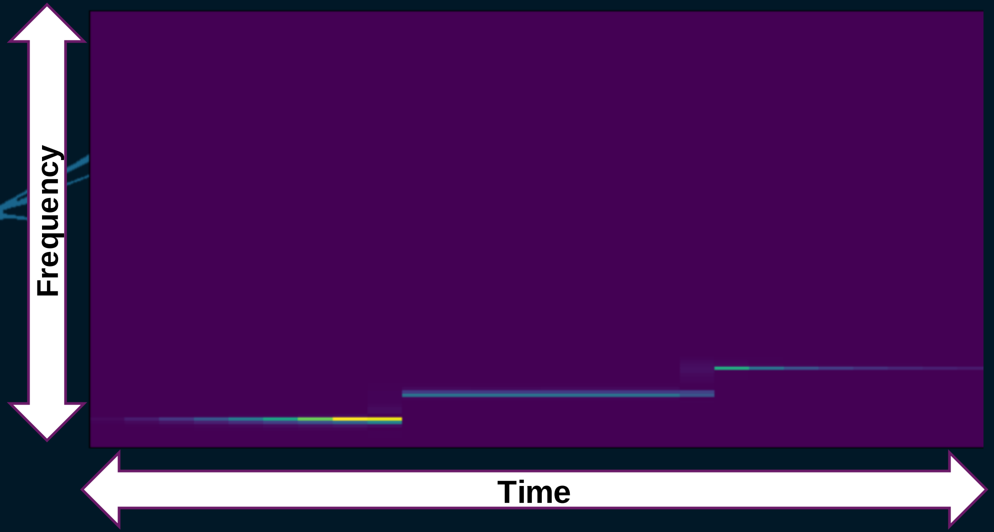
Spectrogram captures this naturally.

But if we use spectrogram, the dimensionality of features will be very large.

Signal in time domain:



Spectrogram view of above signal:

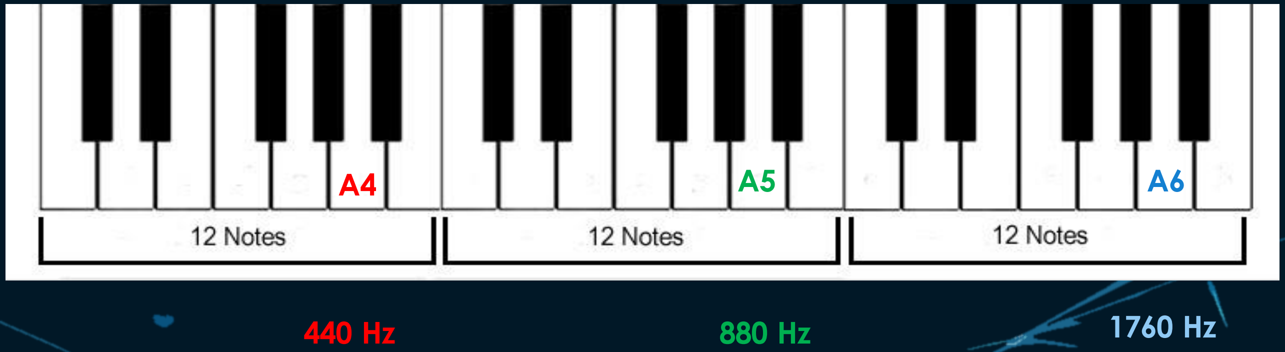


For spectrograms, time resolution and frequency resolution is a tradeoff.

We want Smaller D without losing the higher frequencies.

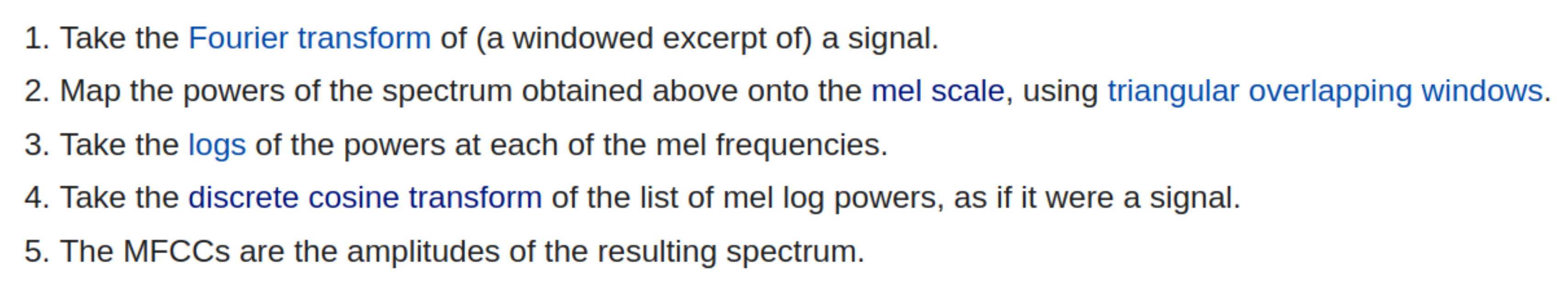
Mel(ody) Spectrogram:

Human perception for audio frequencies and audio gain is non-linear in nature. Spectrogram captures the linear scale. So, MEL-Spectrogram rescales the frequency axis using mel-scale.



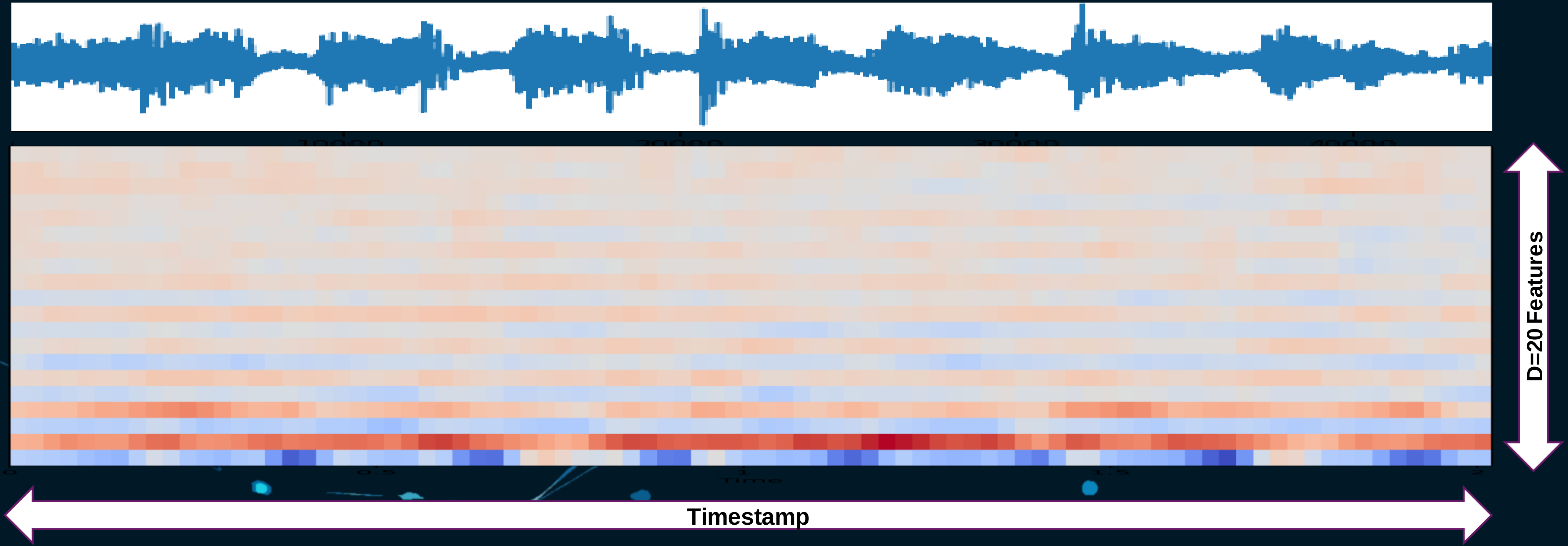
$$F_{mel} = 2595 * \log_{10}\left(1 + \frac{F}{700}\right)$$

Another type of features which has low dimensionality is MFCC features. It emerged from speech processing domain. It is related to Mel-Spectrogram but robust to the noise.



1. Take the Fourier transform of (a windowed excerpt of) a signal.
2. Map the powers of the spectrum obtained above onto the mel scale, using triangular overlapping windows.
3. Take the logs of the powers at each of the mel frequencies.
4. Take the discrete cosine transform of the list of mel log powers, as if it were a signal.
5. The MFCCs are the amplitudes of the resulting spectrum.

MFCC Visualization:

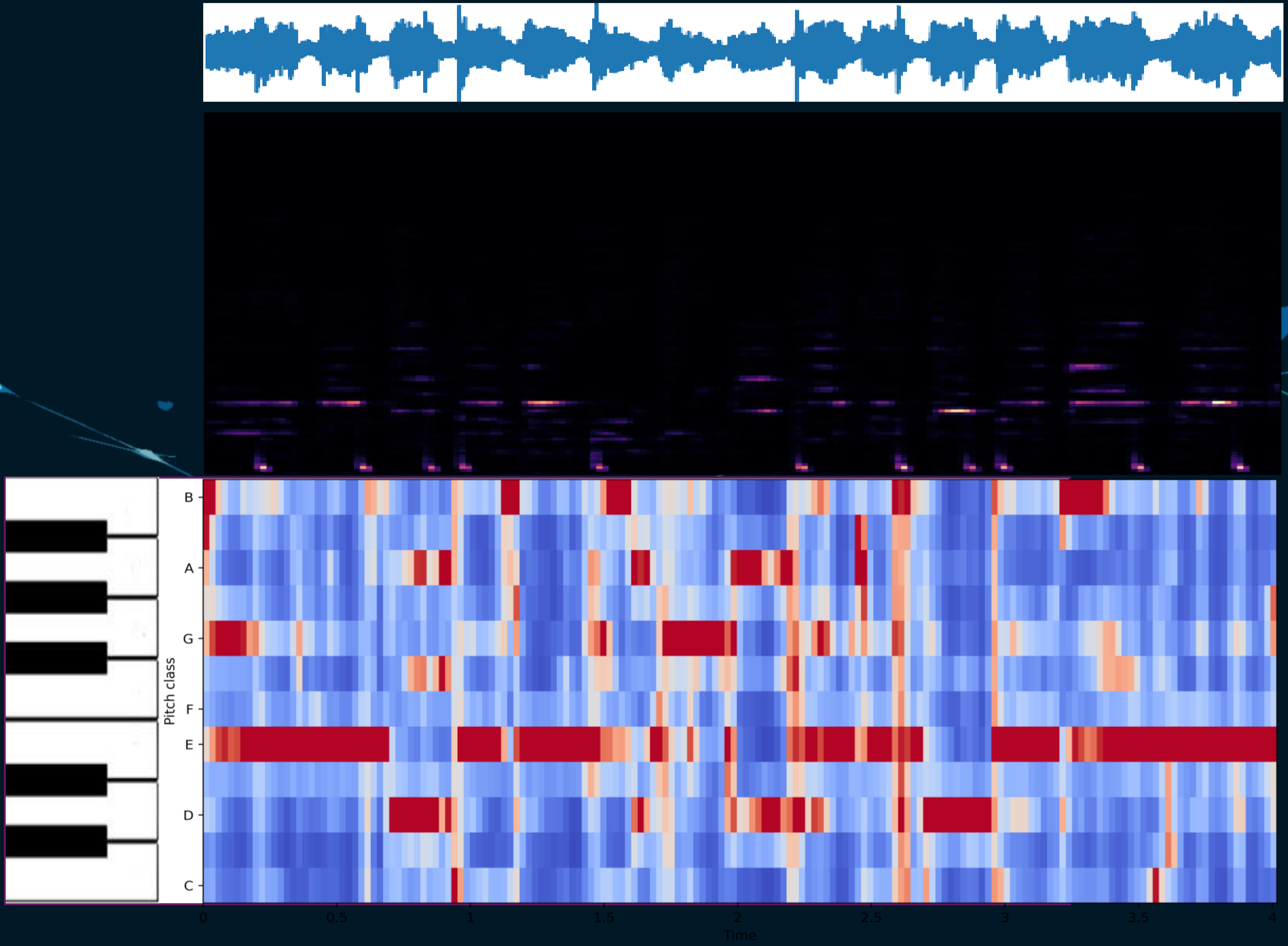


The MFCC Features captures the details about the envelope in IDFT of the Log Power Spectrum.

We will consider 20 MFCC Coefficients.

Chroma Features:

Chroma feature captures the tonal features from local window.



It finds the most dominant tone from 12 tones. So, Per window Its dimensionality is 12.

Spectral Centroid: **It captures the average spectral frequency weighted by intensity.**

$$Spectral\,Centroid\,(Fc) = \frac{\sum_{k=1}^{K} S(x) f(x)}{\sum_{k=1}^{K} S(x)}$$

Its dimensionality is 1 per window.

Spectral Contrast: It considers the spectral peak, the spectral valley, and their difference in each frequency sub band. [Ref: https://musicinformationretrieval.com/spectral_features.html]

Its dimensionality is 7 per window.

Total Number of features per window: (Total No of Windows = No of timestamps)

MFCC: 20,
Chroma-STFT: 12
Spectral Centroid: 1
Spectral Contrast: 7

So, Total features per window = 40.

Dimension of Features for the dataset:

(N, T, D) = (5000, 256, 40)

Length of Each Track: 30 sec
Sample Rate: 22050 Hz
Window Size: 2048 samples
Hop Length: 512 samples

Total Timestamps = (30 * 22050 - 512) / 512 = 1287

We are using LSTM for Sequence Classification It works well for around 400-500 timestamps. So, we will split a track into 5 segments. Now we use all new created segments as additional examples. This is how a kind of Data Augmentation is performed.

After Augmenting the dataset:

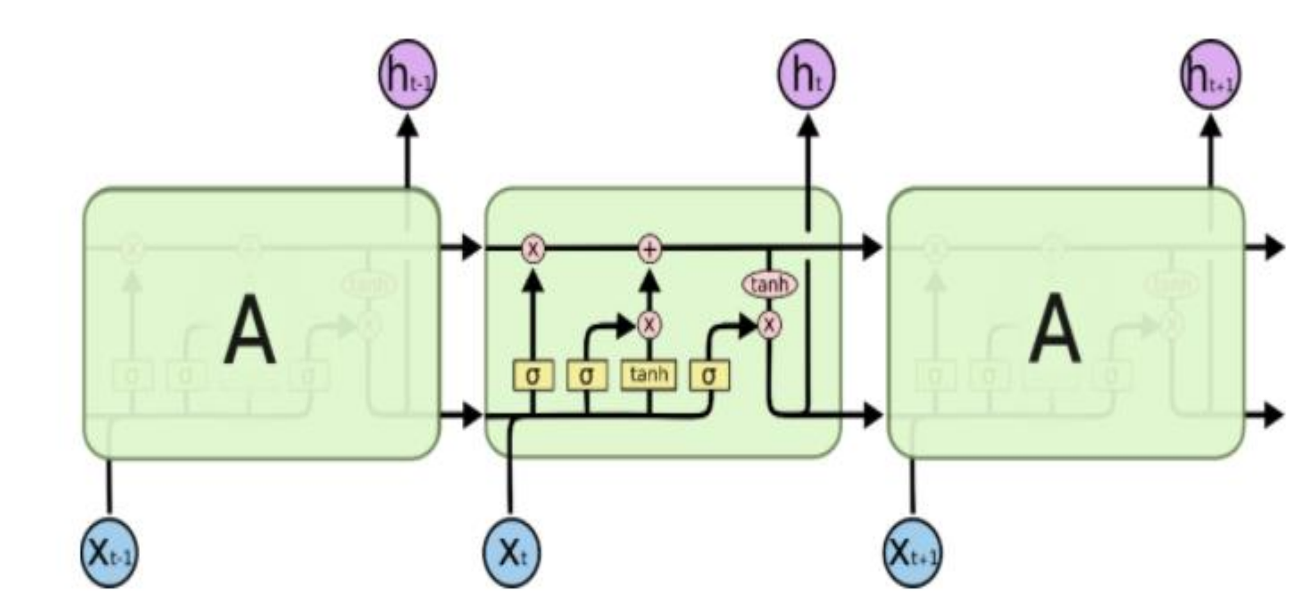
Length of Each Track: 6 sec
Sample Rate: 22050 Hz
Window Size: 2048 samples
Hop Length: 512 samples

Total Timestamps = (6 * 22050 - 512) / 512 = 257

We will select 256 timestamps and ignore the last timestamp to roundoff the sequence length to 256.

LSTM Networks

LSTM are ideal for sequential data, where order is important. Long term dependencies can be preserved with the help of loops, containing four interacting layers.



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

$$C_t = f_t * C_{t-1} + i_t \neq \tilde{C}_t$$

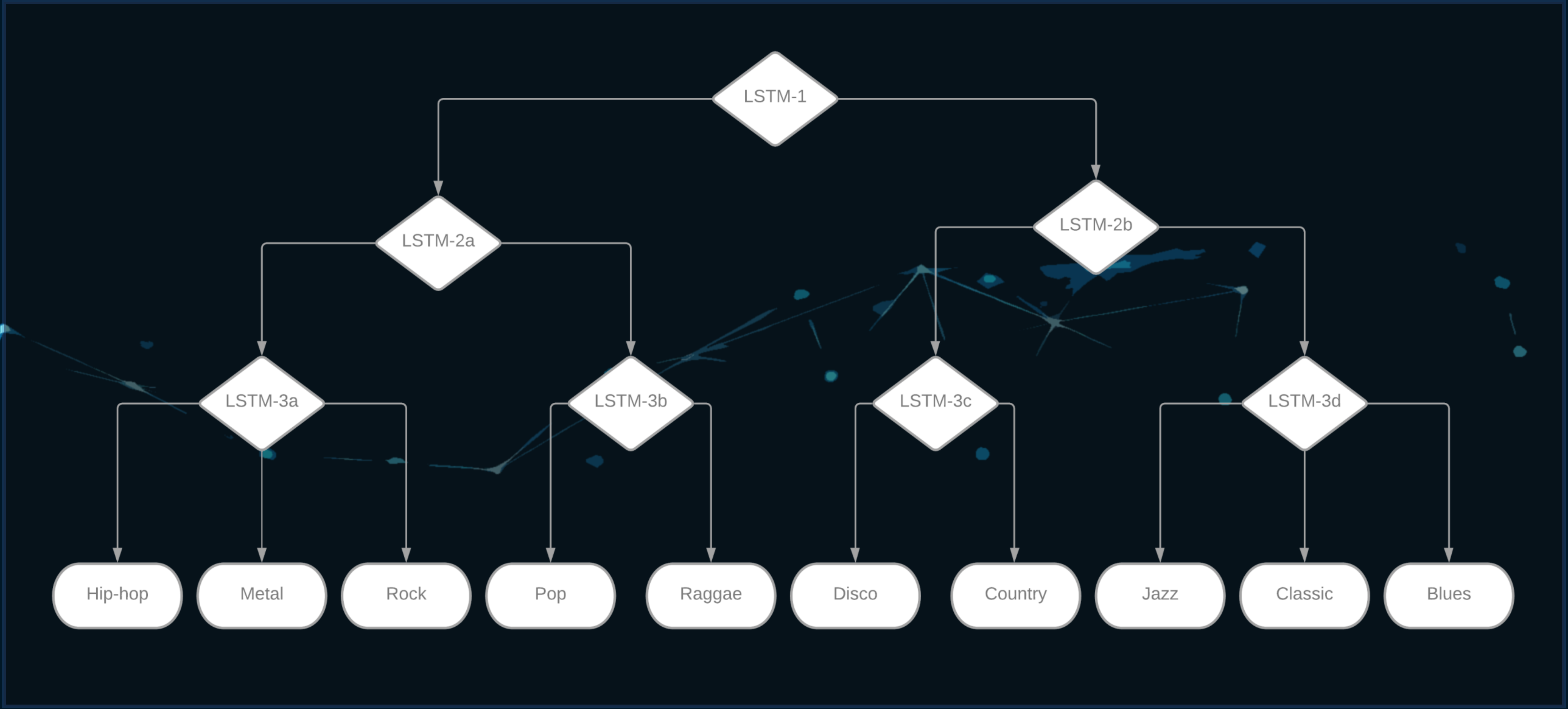
$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

Source: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

The repeating module in an LSTM contains four interacting layers

Hierarchical LSTM Architecture



Adopted from [2] (Tang et al.) with some modifications.

Functionality of the LSTM classifiers

- LSTM1: It classifies between strong (hip-hop, metal, pop, rock, reggae) and mild (jazz, disco, country, classic, blues)
- LSTM2a: It classifies between sub-strong1 (hip-hop, metal, and rock) and sub-strong2 (pop and reggae)
- LSTM2b: It classifies between sub-mild1 (disco and country) and sub-mild2 (jazz, classic, and blues)
- LSTM3a: It classifies between hip-hop, metal and rock
- LSTM3b: It classifies between pop and reggae
- LSTM3c: It classifies between disco and country
- LSTM3d: It classifies between jazz, classic, and blues.

This hierarchical architecture, helps to tackle the multi-class classification problem, by a divide and conquer based approach, where each LSTM in the tree, is trained using samples of the relevant classes. The idea for the hierarchical LSTM model was adopted from the paper [2].

Architecture of the LSTM networks

Input Layer	**20 MFCC, 12 Chroma-STFT, 1 Spectral Centroid, 7 Spectral Contrast, total 40 features obtained as input**
Hidden Layer I	64 LSTM units
Hidden Layer II	32 LSTM units
Output Layer	Number of units depends on which the number of fan-out results. for example, LSTM 3a, will have 3 softmax units whereas LSTM 1 will have 2 softmax units.

Click to add text

Library Used for implementation: Tensorflow

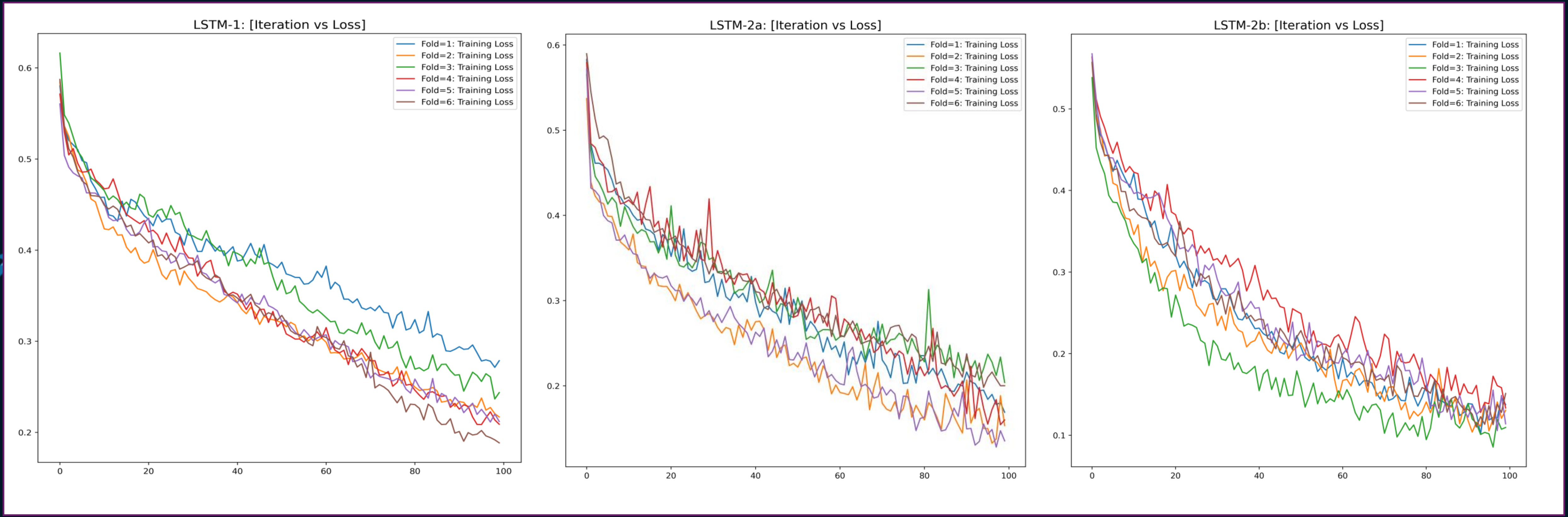
Train, Test, Split:

We are doing K-Fold Cross Validation with K = 6.

	Before Splitting:	After Splitting (Train, Val, Test):			
LSTM 1:	5000	3750	750	500	(Stratified)
LSTM 2a:	2500	1500	300	200	(Stratified*)
LSTM 2b:	2500	1500	300	200	(Stratified*)
LSTM 3a:	1500	1125	225	150	(Stratified)
LSTM 3b:	1000	750	150	100	(Stratified)
LSTM 3c:	1000	750	150	100	(Stratified)
LSTM 3d:	1500	1125	225	150	(Stratified)

* indicates that explicitly Stratified by ignoring some examples. To avoid class imbalance.

Iterations vs Cost Plots:

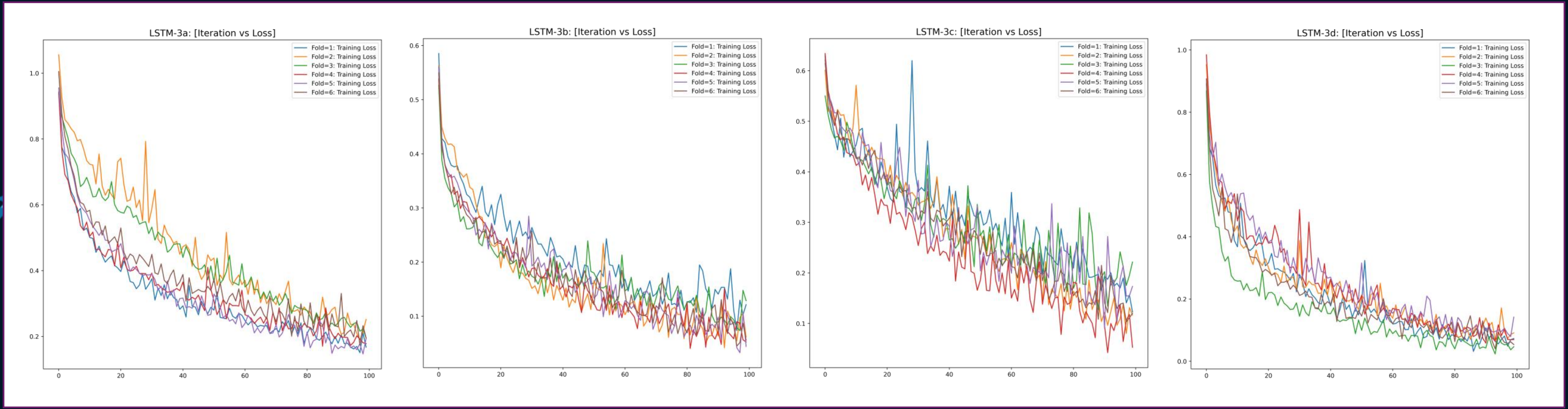


LSTM 1:

LSTM 2a:

LSTM 2b:

Iterations vs Cost Plots: [LSTM 3a, LSTM 3b, LSTM 3c, LSTM 3d]



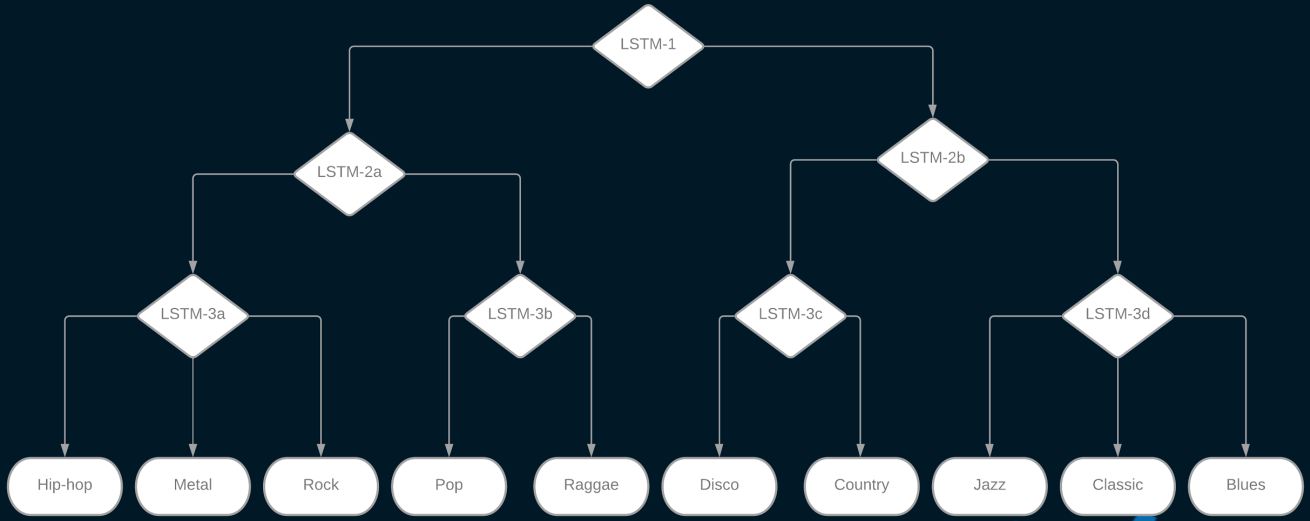
LSTM 3a:

LSTM 3b:

LSTM 3c:

LSTM 3d:

Prediction Methods:



1: Hard Prediction

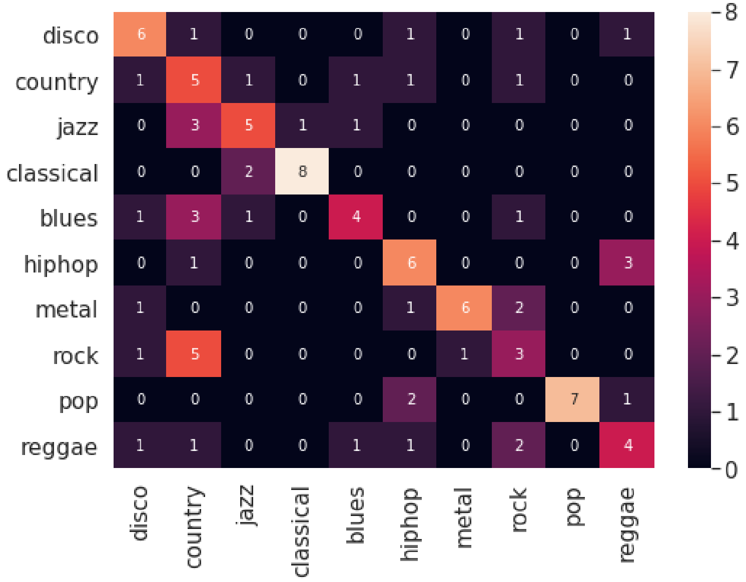
1: In this, we perform segmentation on the original track so that each segment will have 256 timestamps after feature extraction.
2: Predict the class for each segment.
3: Select label with highest frequency.
4: For all segments, select the path with majority predicted label.
5: Repeat for all internal nodes until leaf nodes are not predicted.
6: Return the predicted label.

2: Soft Prediction

1: In this, we perform segmentation on the original track so that each segment will have 256 timestamps after feature extraction.
2: Predict the class for each segment.
3: For each segments, choose path independently.
4: Do this for all internal nodes until leaf nodes are not predicted.
5: Return all predicted labels with the frequencies.

Results Evaluation:

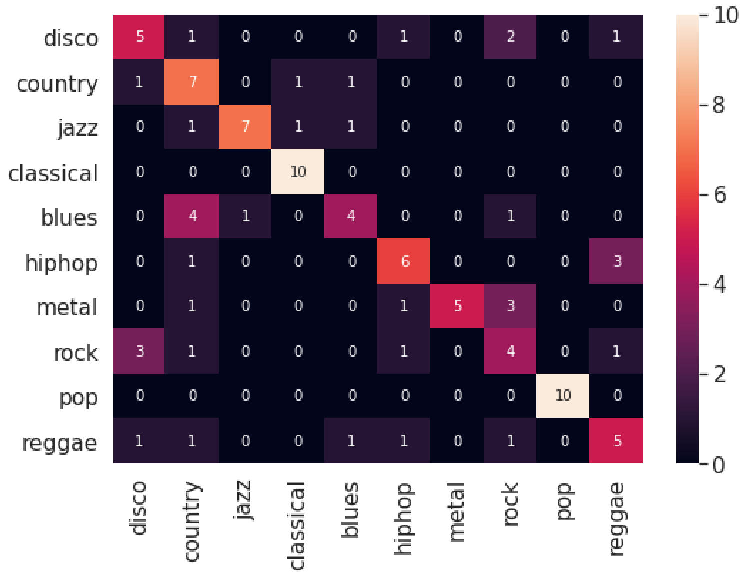
Hard Prediction



	disco	country	jazz	classical	blues	hiphop	metal	rock	pop	reggae
disco	6	1	0	0	0	1	0	1	0	1
country	1	5	1	0	1	1	0	1	0	0
jazz	0	3	5	1	1	0	0	0	0	0
classical	0	0	2	8	0	0	0	0	0	0
blues	1	3	1	0	4	0	0	1	0	0
hiphop	0	1	0	0	0	6	0	0	0	3
metal	1	0	0	0	0	1	6	2	0	0
rock	1	5	0	0	0	0	1	3	0	0
pop	0	0	0	0	0	2	0	0	7	1
reggae	1	1	0	0	1	1	0	2	0	4

Accuracy: 54%

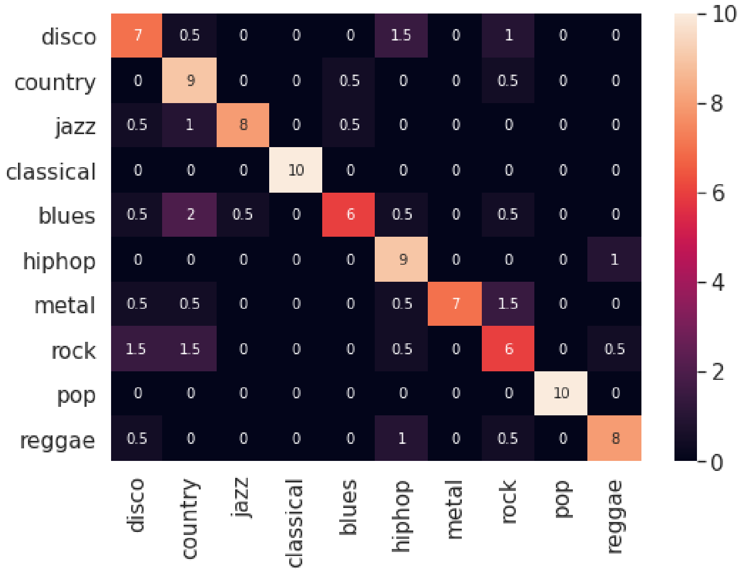
Soft Prediction (Top-1)



	disco	country	jazz	classical	blues	hiphop	metal	rock	pop	reggae
disco	5	1	0	0	0	1	0	2	0	1
country	1	7	0	1	1	0	0	0	0	0
jazz	0	1	7	1	1	0	0	0	0	0
classical	0	0	0	10	0	0	0	0	0	0
blues	0	4	1	0	4	0	0	1	0	0
hiphop	0	1	0	0	0	6	0	0	0	3
metal	0	1	0	0	0	1	5	3	0	0
rock	3	1	0	0	0	1	0	4	0	1
pop	0	0	0	0	0	0	0	0	10	0
reggae	1	1	0	0	1	1	0	1	0	5

Accuracy: 63%

Soft Prediction (Top-2)



	disco	country	jazz	classical	blues	hiphop	metal	rock	pop	reggae
disco	7	0.5	0	0	0	1.5	0	1	0	0
country	0	9	0	0	0.5	0	0	0.5	0	0
jazz	0.5	1	8	0	0.5	0	0	0	0	0
classical	0	0	0	10	0	0	0	0	0	0
blues	0.5	2	0.5	0	6	0.5	0	0.5	0	0
hiphop	0	0	0	0	0	9	0	0	0	1
metal	0.5	0.5	0	0	0	0.5	7	1.5	0	0
rock	1.5	1.5	0	0	0	0.5	0	6	0	0.5
pop	0	0	0	0	0	0	0	0	10	0
reggae	0.5	0	0	0	0	1	0	0.5	0	8

Accuracy: 80%

As we can see that using hard prediction method, accuracy is as good as mentioned in [2]. Using soft prediction method, it improved from 53 % to 63 %. And using top-2 predictions it is improved to 80%.

Tools and Libraries Used:

1: Librosa (To extract features)

2: TensorFlow (To create LSTM models)

3: Matplotlib (To create plots)

References:

- G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," in IEEE Transactions on Speech and Audio Processing, vol. 10, no. 5, pp. 293-302, July 2002, doi: 10.1109/TSA.2002.800560. [1]

- Tang, C. P., Chui, K. L., Yu, Y. K., Zeng, Z., & Wong, K. H. (2018, July). Music genre classification using a hierarchical long short-term memory (LSTM) model. In *Third International Workshop on Pattern Recognition* (Vol. 10828, p. 108281B). International Society for Optics and Photonics. [2]

Thank You:

Spotify tells me that i have listened to 941 different genres
me thinking there only exist 6